CENG 329

Microprocessors

2024-2025 Fall

Semester Project Report

Names of members

Çağdaş GÜLEÇ 202111004

Doğukan POYRAZ 202111061

Arda YILDIZ 202111071

Barkın SARIKARTAL 202111039

Date

04.01.2025

YouTube Link: https://youtu.be/0pIQgclVjL4?si=2iatSyHiZ5BjECWG

Table of Contents

1. Introduction

This report examines a game project designed with an MSP430 microcontroller, written in assembly language. The project uses a 7-segment display, two buttons, and two LEDs to create a simple competitive game scenario. The main goal is to count down from 3 to 0 on the 7-segment display and determine the winner based on when the players press their buttons (either before or after the countdown reaches 0). This report explains the hardware components used, the flow of the code, the interrupt structures, and the overall operating principle of the system step by step.

2. Project Objectives and Summary

The objective is to create a game that counts down from 3 to 0 on a 7-segment display and uses two buttons and two LEDs to implement the following rules:

1. **Countdown Mechanism:**
   - The 7-segment display counts down 3, 2, 1, 0 at one-second intervals.
   - Once the countdown reaches 0, the first player to press the button wins.
2. **Win Conditions:**
   - If a button is pressed before the countdown reaches 0, the other player automatically wins.
   - If both players wait until 0 and then press, the first to press wins.
3. **LED Indicators:**
   - P2.2 and P2.3 are LEDs indicating which player has won.
   - The winning LED turns on briefly to show the result.
4. **Game Restart:**
   - After a win, the game waits 3 seconds and then restarts automatically.
5. **Bonus / Additional Features:**
   - Manual reset mechanism (optional).
   - Implementing the project exclusively with interrupts (for extra points).

These rules are realized using the MSP430’s I/O pins, interrupts, and simple timing delays.

3. Hardware Components and Connections

Below is a summary of the main hardware components used in this project and their corresponding port/pin assignments:

1. **MSP430 Microcontroller (e.g., MSP430G2553):**
    - Port 1: Connected to the 7-segment display's segments.
    - Port 2: Two input buttons (P2.0, P2.1) and two LEDs (P2.2, P2.3).
2. **7-Segment Display:**
    - P1.0: Segment a
    - P1.1: Segment b
    - P1.2: Segment c
    - P1.4: Segment d
    - P1.5: Segment e
    - P1.6: Segment f
    - P1.7: Segment g
3. **Buttons:**
    - P2.0 and P2.1: Player input buttons.
4. **LEDs:**
    - P2.2: LED indicating player one.
    - P2.3: LED indicating player two.

4. Code Structure and Flow

The code is written in MSP430 Assembly and consists of the following main blocks:

- .cdecls and RESET definitions
- Memory Setup (Stack Pointer, stopping the Watchdog Timer)
- Pin/Port Configuration
- Main Loop (MainLoop)
- Delay Functions (delay1Sec, delay3Sec)
- 7-Segment Display Functions (3, 2, 1, 0, and blank/dash)
- Control Function (controlFunction)
- Interrupt Service Routines (for Port 1 and Port 2)

Below is a detailed explanation of each part.

4.1. Memory and Watchdog Setup

```
RESET       mov.w   #__STACK_END,SP         ; Initialize Stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop Watchdog Timer
```

- Stack Pointer is initialized to the end of RAM.

- Watchdog Timer is stopped to prevent the MCU from resetting unexpectedly.

4.2. Pin I/O and Interrupt Configuration

```
;-------------------------------------------------------------------------------
;                         Main Loop and Pin Definitions
;-------------------------------------------------------------------------------
;
;            P1.0, 1.1, 1.2, 1.4, 1.5, 1.6, 1.7 => 7 Segment LED Pins
;            P2.0, 2.1 => Buttons
;            P2.2, 2.3 => LEDs
;
;-------------------------------------------------------------------------------

;--- Definitions and Initial Pin Setup ---
      bic.b #11110111b, &P1SEL     ; Clear P1SEL to Use Port 1 as GPIO
      bic.b #11110111b, &P1SEL2
      bic.b #00001111b, &P2SEL     ; Clear Lower 4 Bits of P2SEL for GPIO
      bic.b #00001111b, &P2SEL2

; 7 Segments And LEDs Output Setup
      bis.b #11110111b, &P1DIR     ; Set Corresponding Pins in P1DIR as Output
      bis.b #00001100b, &P2DIR

; Buttons Setup
      bic.b #00000011b, &P2DIR     ; P2.0, P2.1 Input for Buttons
      bis.b #00000011b, &P2REN     ; Enable Pull-Up Resistors
      bis.b #00000011b, &P2OUT

; Clear Outputs
      bic.b #11110111b, &P1OUT     ; Turn Off All 7-Segment LED Pins Initially
      bic.b #00001100b, &P2OUT     ; Turn Off Both LEDs Initially

; Enable Global Interrupts
      bis.w #GIE, SR               ; General Interrupt Enable

; Configure Interrupt Edge for Buttons on Port 2.0,2.1 H to L
      bis.b #00000011b, &P2IES
```

Key points:

- P1 pins are configured as outputs to drive the 7-segment display segments.
- P2.2 and P2.3 are outputs for LEDs.
- P2.0 and P2.1 are inputs with pull-up resistors for the buttons.
- The interrupt trigger is set for a falling edge (High to Low) transition.

4.3. Main Loop (MainLoop)

```
;-------------------------------------------------------------------------------
;                                      Main
;-------------------------------------------------------------------------------
mainLoop:
      bis.b #00000011b, &P2IE      ; Enable Interrupts for P2.0 and P2.1

      call #sevenSegments_3
      call #controlFunction
      call #delay1Sec
      call #controlFunction

      call #sevenSegments_2
      call #controlFunction
      call #delay1Sec
      call #controlFunction

      call #sevenSegments_1
      call #controlFunction
      call #delay1Sec
      call #controlFunction

      call #sevenSegments_0
      call #controlFunction
      call #delay1Sec
      call #controlFunction

      bic.b #00000011b, &P2IE      ; Disable Interrupts for P2.0 and P2.1
      call #sevenSegments_Blank
      call #delay3Sec

      bic.b #00000011b, &P2IFG     ; Clear Port 2 Interrupt Flags
      jmp mainLoop                 ; Repeat the Main Loop Indefinitely
```

This loop manages the 3 -> 0 countdown on the 7-segment:

1. Displays 3, 2, 1, 0 in order, each with a 1-second delay.
2. After reaching 0, it disables button interrupts, displays a “dash” (or single segment) on the 7-segment, then waits 3 seconds.
3. Clears the interrupt flags and jumps back to MainLoop, repeating indefinitely.

Hence, the program continuously performs a countdown and re-enables the button interrupts during the countdown.

4.4. Delay Functions

The code includes two delay functions:

- **delay1Sec:** Roughly 1-second delay.
- **delay3Sec:** Roughly 3-second delay.

```
;--- Delay 1 Second ---
delay1Sec:
      mov.w #0 , r4
      mov.w #0 , r5


delay_outer1sec:
      add.w #1 , r4
```

```
delay_inner1sec:
      add.w #1 , r5
      cmp #50000 , r5
      jne delay_inner1sec
      cmp #4 , r4
      jne delay_outer1sec
      ret
```

- Nested loops (using r4 and r5) increment until a certain count is reached, creating the delay.
- Once the loops are complete, the function returns.

Similarly, delay3Sec uses the same logic with a different loop range to produce a longer delay.

4.5. 7-Segment Display Functions

The code defines separate functions for displaying 3, 2, 1, 0, and blank on the 7-segment. Each function sets the correct combination of Port 1 bits to turn on the required segments.

For example, “3” is displayed with sevensegments_3:

```
;--- Display Number "3" on 7 Segment LED ---
sevenSegments_3:                       ; Turn On 7 Segment LED's a, b, c, d, g LEDs
      mov.w #3 , r6                    ; Store Digit '3' in r6
      bic.b #11110111b, &P1OUT
      bis.b #10010111b, &P1OUT
      ret
```

Each digit function activates a different combination of segments. The blank function turns off all segments except for segment g (showing a dash in the center).

4.6. Control Function (controlFunction)

```
;--- Control Function Checks State in r6 ---
controlFunction:   ; Controller Function of Restarting Game When the Game Ends
      cmp #-2 , r6
      jeq mainLoop                    ; If r6 == -2, Jump Back to mainLoop
      ret
```

This function checks the r6 register value to decide whether to jump back to MainLoop (when r6 == -2) or simply return. In the interrupt service routine, when mov.w #-2, r6 is executed, the code returns to MainLoop on the next call to controlFunction, signaling the game has ended.

5. Interrupt Service Routines

Most of the core game logic is contained in the Port 1 and Port 2 interrupt routines.

5.1. Port 2 Interrupt (Game)

;--- Button Game Interrupt (Port 2) ---

```
Game:
        cmp #0, r6                      ; Current LED Position When Pressed
        jeq State_1

State_2:                                ; Function Applied for the Baseman Before the Game Starts
        bic.b #00000011b, &P2IE         ; Disable Interrupts on P2.0 and P2.1
        bit.b #00000001b, &P2IN         ; Check P2.0 Button State
        jeq player2_earlyPress

player1_earlyPress:
        bis.w #00001000b , &P2OUT
        call #delay1Sec
        bic.w #00001000b , &P2OUT
        jmp out

player2_earlyPress:
        bis.w #00000100b , &P2OUT
        call #delay1Sec
        bic.w #00000100b , &P2OUT
        jmp out

State_1:                                 ; Control Operation When Pressed on Time
        bic.b #00000011b, &P2IE
        bit.b #00000001b, &P2IN
        jeq player2_win

player1_win:
        bis.w #00000100b , &P2OUT
        call #delay1Sec
        bic.w #00000100b , &P2OUT
        jmp out

player2_win:
        bis.w #00001000b , &P2OUT
        call #delay1Sec
        bic.w #00001000b , &P2OUT

out:
        bis.b #00000011b, &P2IE         ; Re-enable Interrupts
        bic.b #00000011b, &P2IFG        ; Clear Interrupt Flags for P2.0 and P2.1
        mov.w #-2 , r6
```

- The code checks if r6 equals 0 (meaning the countdown has reached 0) or not.
- In State_2 (before countdown is 0), if a player presses their button, the other player's LED is lit to indicate the pressing player lost.
- In State_1 (when r6 = 0), the first player to press gets their LED lit, indicating they have won.
- The code disables interrupts, checks which button is pressed (bit.b #00000001b, &P2IN for P2.0), lights the correct LED, calls delay1sec, and then turns the LED off.
- Finally, it re-enables interrupts and sets r6 to -2 to indicate the game has ended, causing a return to MainLoop.

In short:

- Pressing the button before 0 means you lose and the other LED turns on.
- Pressing the button right when 0 means you win if you're the first to press.
- Players cannot press button after 7-segment displays -.

6. Conclusion and Evaluation

This project demonstrates a simple competitive game using the MSP430 microcontroller in assembly language. The key points are:

1. **Countdown:** 7-segment display shows 3, 2, 1, 0.
2. **Button Logic:**
   - If a button is pressed before 0, the other player wins.
   - If both wait until 0, the first to press wins.
3. **LED Feedback:** The winning player's LED lights up for one second.
4. **Automatic Restart:** The game waits three seconds after a round and restarts.

Appendix

```
;-------------------------------------------------------------------------------
;       MSP430 Assembler Code Template for Use With TI Code Composer Studio
;
;-------------------------------------------------------------------------------
            .cdecls C,LIST,"msp430.h"       ; Include Device Header File

;-------------------------------------------------------------------------------
            .def    RESET                   ; Export Program Entry-Point to
                                            ; Make It Known to Linker.
;-------------------------------------------------------------------------------
            .text                           ; Assemble Into Program Memory.
            .retain                         ; Override ELF Conditional Linking
                                            ; and Retain Current Section.
            .retainrefs                     ; And Retain Any Sections That Have
                                            ; References to Current Section.

;-------------------------------------------------------------------------------
RESET       mov.w   #__STACK_END,SP         ; Initialize Stackpointer
StopWDT     mov.w   #WDTPW|WDTHOLD,&WDTCTL  ; Stop Watchdog Timer

;-------------------------------------------------------------------------------
;                         Main Loop and Pin Definitions
;-------------------------------------------------------------------------------
;
;            P1.0, 1.1, 1.2, 1.4, 1.5, 1.6, 1.7 => 7 Segment LED Pins
;            P2.0, 2.1 => Buttons
;            P2.2, 2.3 => LEDs
;
;-------------------------------------------------------------------------------

;--- Definitions and Initial Pin Setup ---
      bic.b #11110111b, &P1SEL     ; Clear P1SEL to Use Port 1 as GPIO
      bic.b #11110111b, &P1SEL2
      bic.b #00001111b, &P2SEL     ; Clear Lower 4 Bits of P2SEL for GPIO
      bic.b #00001111b, &P2SEL2

; 7 Segments And LEDs Output Setup
      bis.b #11110111b, &P1DIR     ; Set Corresponding Pins in P1DIR as Output
      bis.b #00001100b, &P2DIR

; Buttons Setup
      bic.b #00000011b, &P2DIR     ; P2.0, P2.1 Input for Buttons
      bis.b #00000011b, &P2REN     ; Enable Pull-Up Resistors
      bis.b #00000011b, &P2OUT

; Clear Outputs
      bic.b #11110111b, &P1OUT     ; Turn Off All 7-Segment LED Pins Initially
      bic.b #00001100b, &P2OUT     ; Turn Off Both LEDs Initially

; Enable Global Interrupts
      bis.w #GIE, SR               ; General Interrupt Enable

; Configure Interrupt Edge for Buttons on Port 2.0,2.1 H to L
      bis.b #00000011b, &P2IES
```

```
;-------------------------------------------------------------------------------
;                                      Main
;-------------------------------------------------------------------------------
mainLoop:
      bis.b #00000011b, &P2IE       ; Enable Interrupts for P2.0 and P2.1

      call #sevenSegments_3
      call #controlFunction
      call #delay1Sec
      call #controlFunction

      call #sevenSegments_2
      call #controlFunction
      call #delay1Sec
      call #controlFunction

      call #sevenSegments_1
      call #controlFunction
      call #delay1Sec
      call #controlFunction

      call #sevenSegments_0
      call #controlFunction
      call #delay1Sec
      call #controlFunction

      bic.b #00000011b, &P2IE       ; Disable Interrupts for P2.0 and P2.1
      call #sevenSegments_Blank
      call #delay3Sec

      bic.b #00000011b, &P2IFG      ; Clear Port 2 Interrupt Flags
      jmp mainLoop                  ; Repeat the Main Loop Indefinitely

;-------------------------------------------------------------------------------
;                                      Functions
;-------------------------------------------------------------------------------

;--- Delay 3 Seconds ---
delay3Sec:
      mov.w #0 , r4                 ; Initialize Counter r4
      mov.w #0 , r5                 ; Initialize Counter r5

delay_outer3sec:
      add.w #1 , r4                 ; Outer Loop Increment

delay_inner3sec:
      add.w #1 , r5                 ; Inner Loop Increment
      cmp #50000 , r5               ; Compare r5 with 50000
      jne delay_inner3sec           ; If not Equal, Keep Looping
      cmp #12 , r4                  ; Compare r4 with 12
      jne delay_outer3sec           ; If not Equal, Keep Looping
      ret                           ; Return when Both Loops Complete

;--- Delay 1 Second ---
delay1Sec:
      mov.w #0 , r4
      mov.w #0 , r5
```

```
delay_outer1sec:
       add.w #1 , r4

delay_inner1sec:
       add.w #1 , r5
       cmp #50000 , r5
       jne delay_inner1sec
       cmp #4 , r4
       jne delay_outer1sec
       ret

;--- Display Number "3" on 7 Segment LED ---
sevenSegments_3:                     ; Turn On 7 Segment LED's a, b, c, d, g LEDs
       mov.w #3 , r6                    ; Store Digit '3' in r6
       bic.b #11110111b, &P1OUT
       bis.b #10010111b, &P1OUT
       ret

;--- Display Number "2" on 7 Segment LED ---
sevenSegments_2:                     ; Turn On 7 Segment LED's a, b, d, e, g LEDs
       mov.w #2 , r6
       bic.b #10010111b, &P1OUT
       bis.b #10110011b, &P1OUT
       ret

;--- Display Number "1" on 7 Segment LED ---
sevenSegments_1:                     ; Turn On 7 Segment LED's b and c LEDs
       mov.w #1 , r6
       bic.b #10110011b, &P1OUT
       bis.b #00000110b, &P1OUT
       ret

;--- Display Number "0" on 7 Segment LED ---
sevenSegments_0:                     ; Turn On 7 Segment LED's a, b, d, e, f LEDs
       mov.w #0 , r6
       bic.b #00000110b, &P1OUT
       bis.b #01110111b, &P1OUT
       ret

;--- Turn All Segments Off and Turn On 7 Segment LED'S g LED ---
sevenSegments_Blank:
       mov.w #-1 , r6
       bic.b #01110111b, &P1OUT
       bis.b #10000000b, &P1OUT
       ret

;--- Control Function Checks State in r6 ---
controlFunction:                         ; Controller Function of Restarting Game
When the Game Ends
       cmp #-2 , r6
       jeq mainLoop                  ; If r6 == -2, Jump Back to mainLoop
       ret
```

```
;-------------------------------------------------------------------------------
;                                      Interrupts
;-------------------------------------------------------------------------------

;--- Button Game Interrupt (Port 2) ---
Game:
      cmp #0, r6                               ; Current LED Position When Pressed
      jeq State_1

State_2:                                       ; Function Applied for the Baseman
Before the Game Starts
      bic.b #00000011b, &P2IE       ; Disable Interrupts on P2.0 and P2.1
      bit.b #00000001b, &P2IN       ; Check P2.0 Button State
      jeq player2_earlyPress

player1_earlyPress:
      bis.w #00001000b , &P2OUT
      call #delay1Sec
      bic.w #00001000b , &P2OUT
      jmp out

player2_earlyPress:
      bis.w #00000100b , &P2OUT
      call #delay1Sec
      bic.w #00000100b , &P2OUT
      jmp out

State_1:                                       ; Control Operation When Pressed on
Time
      bic.b #00000011b, &P2IE
      bit.b #00000001b, &P2IN
      jeq player2_win

player1_win:
      bis.w #00000100b , &P2OUT
      call #delay1Sec
      bic.w #00000100b , &P2OUT
      jmp out

player2_win:
      bis.w #00001000b , &P2OUT
      call #delay1Sec
      bic.w #00001000b , &P2OUT

out:
      bis.b #00000011b, &P2IE       ; Re-enable Interrupts
      bic.b #00000011b, &P2IFG      ; Clear Interrupt Flags for P2.0 and P2.1
      mov.w #-2 , r6
      reti

;-------------------------------------------------------------------------------
;                                Stack Pointer Definition
;-------------------------------------------------------------------------------
            .global __STACK_END
            .sect   .stack
```

```
;-------------------------------------------------------------------------------
;                                  Interrupt Vectors
;-------------------------------------------------------------------------------

             .sect ".int03"                   ; Port 2 Interrupt Vector
             .short Game
             .sect   ".reset"                 ; MSP430 RESET Vector
             .short  RESET
```

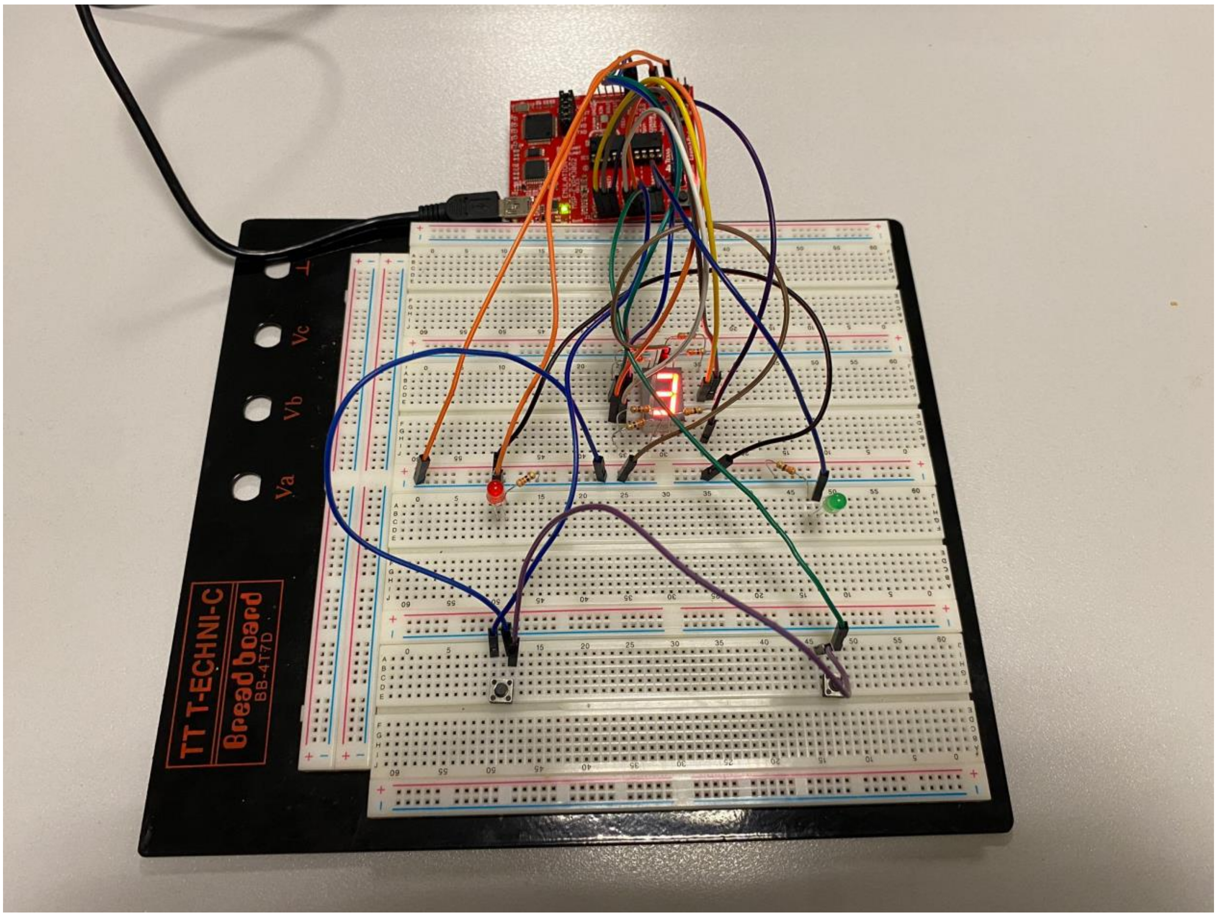

Figure 1. Front Angle Image of Breadboard Circuit for The Project

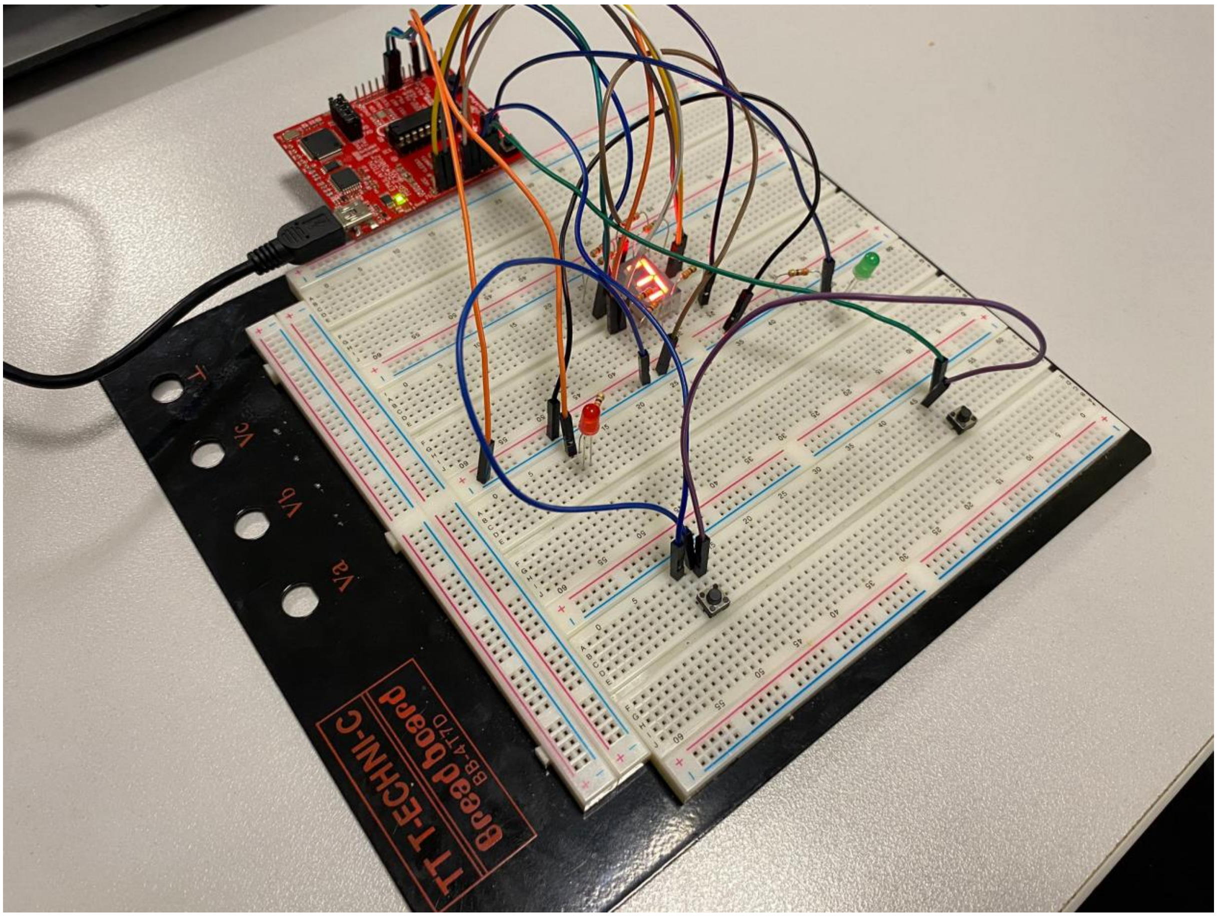

Figure 2. Diagonal Angle Image of Breadboard Circuit for The Project